Report

for Project Stereo

repo@ https://github.com/QYHcrossover/Project_Stereo

emai@qinyuheng@hhu.edu.cn

by 秦瑜恒

目录

另自己写的博客：

双目相机基本概念

立体匹配算法笔记

Overview

第一次接触立体深度估计这方面的项目，之前计算机视觉了解得多的就只有图像识别、目标检测、语义分割这个三个基本的任务了。

应该来说前后花了不到 7 天时间，从单相机标定开始，逐步了解到双相机标定，立体校正，立体匹配算法，最后到双目视觉和单目视觉的深度学习方法，自我感觉学习进度蛮快的了，自己心里也有点小得意。在作业和编程的完成情况如下：

- 作业方面：除了 20 题用 CUDA 加速 SGM 算法，23 题自己加速一种深度学习方法，以及 19 题比较当下最好的算法和 SGM 算法未完成外，其他题目都有作答(尽自己全力去答了)
- 编程方面：单相机标定，双相机标定、畸变校正、立体校正等自己使用 python-opencv 实现完成，跑通代码(结果也应该问题不大吧)；张正友算法底层实现这部分从网上找到了一份源码，也能跑通(跑出来结果和 opencv 的 API 实现的结果差不多)
- 概念方面：
  - 前面单相机标定，双相机标定，立体校正相关概念理解得应该还算透彻吧，题目中的一些公式推导题也都做出来了，不过也不知道具体正确与否
  - 立体匹配算法方面，我看了一个技术讲座；大概了解了立体匹配的基本内容，包括难点、匹配流程(代价计算方法、代价空间、代价聚合……）等
  - 最后关于深度学习下的双目视觉和单目视觉，找了很多综述性的文章来看，也是有了一些基本的概念（21,22 题），尝试地用云 GPU 跑了下 PSM-Net 的测试代码，感觉还行
- 其他：
  - 写了两篇博客，第一篇双目相机基本概念；是看了 tutorial2 这个讲义后，对极线约束、本征矩阵、基础矩阵这几个概念简直豁然开朗啊，这份讲义图真的很好，也有基本的推导公式在上面。
  - 另外一篇是立体匹配算法笔记；是看了奥比中光这个企业的一个技术讲座做的笔记，讲了很多立体匹配的基本概念

这几天基本上每时每刻都投入到了这个项目当中去，一方面是想借这个项目好好表现自己，**希望证明自己的能力**；另一方面**探索知识，解决问题**这个过程也像打怪升级一样是容易上瘾的。通过这个 project，我也对这块产生了浓厚的兴趣，甚至萌生出一个毕业设计搞个树莓派装俩摄像头试试能不能测量深度距离啊，哈哈哈，如果有条件的就试试。

Q1: About intrinsics and extrinsics of a camera and camera matrix

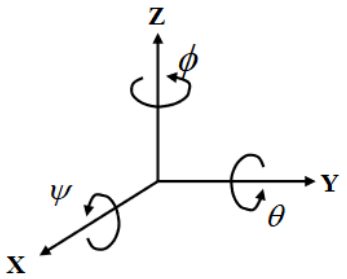
Ans 1:

相机的内参是相机本身的参数，包括 **焦距 focal length**, **image sensor format** 缩放大小, **principal point 像主点坐标**；相机的内参是相机的**固有参数**，每次拍摄相机的内参都是一样的；相机的内参其实构成了由**相机坐标系 camera coordinate** 到**像素坐标系 pixel coordinate** 的内参矩阵 **K**,完成的是一种**透视变换和仿射变换**

$$K = \begin{bmatrix} \alpha_x & \gamma & u_0 & 0 \\ 0 & \alpha_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

相机的外参每次拍摄都不相同，主要和拍摄的位置和角度有关；像素的外参构成了**外参矩阵** extrinsic matrix，它对应的是由**真实世界的坐标系 world coordinates** 到**相机坐标系 camera coordinate** 的变换，它是一种**刚体变换**；这种变换可以由三维**旋转**和**平移**得到，所以相机的外参主要就是旋转和平移的参数，可以表示为 R1，R2，R3 和 T

$$\mathbf{x} \sim \mathbf{K}\underbrace{[\mathbf{R}|\mathbf{t}]}_{\text{外参矩阵}}\mathbf{X}$$

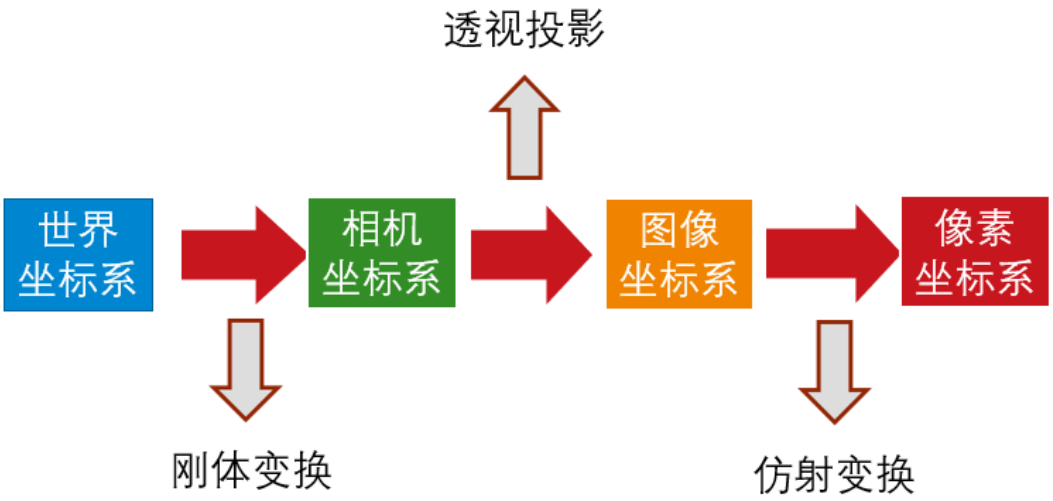


$$R = Rot(z,\varphi)Rot(y,\theta)Rot(x,\psi)$$

$$= \begin{bmatrix} C\varphi & -S\varphi & 0 \\ S\varphi & C\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} C\theta & 0 & S\theta \\ 0 & 1 & 0 \\ -S\theta & 0 & C\theta \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & C\psi & -S\psi \\ 0 & S\psi & C\psi \end{bmatrix}$$

$$= \begin{bmatrix} C\varphi C\theta & -S\varphi S\psi + C\varphi S\theta S\psi & C\varphi S\theta C\psi + S\varphi S\psi \\ S\varphi C\theta & C\varphi S\theta S\psi + C\varphi S\psi & -C\varphi S\psi + S\varphi S\theta C\psi \\ -S\theta & C\theta S\psi & C\theta C\psi \end{bmatrix}$$

camera matrix 表示的就是将由外参决定的刚体变换和内参决定的透视变换和仿射变换合并为一种变换，这种变换的对应的矩阵就是 **camera matrix**,附一张由**世界坐标系**变换到**相机坐标系**，再到**图像坐标系**最后到**像素坐标系**的过程；



$$Z\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{dX} & -\frac{cot\theta}{dX} & u_0 \\ 0 & \frac{1}{dYsin\theta} & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix} \begin{pmatrix} U \\ V \\ W \\ 1 \end{pmatrix}$$

仿射变换　　透视投影　　刚体变换

内参矩阵　　外参矩阵

References：

- CSDN|相机标定的原理及实现
- wikipedia|Camera_resectioning
- 知乎|相机标定之张正友标定法数学原理详解

Q2: Transform a 3D point onto the image plane

Ans 2:

Intrinstic Matrix:

$$K=\begin{bmatrix} f_x & 0 & C_x & 0 \\ 0 & f_y & C_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Extrinsic Matrix:

$(R|t)$

3D point (要齐次)

$$X=\begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ 1 \end{pmatrix}$$

answer is $K\cdot(R|t)X$

Q3: 2D image to 3D camera coordinate

Ans3:

set M = camera matrix

equation: $M\cdot\begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ 1 \end{pmatrix}=\begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$

Q4: About distortion

1. **畸变 distortion** 发生在**摄像机坐标系到图像坐标系变换**时，在透视变换时由于一些因素导致图像发生畸变，引起原始图像的失真；畸变主要分为**径向畸变**和**切向畸变**两种；

   - **径向畸变**主要是镜头的制造工艺不足，是透镜本身的原因导致的；主要又分为桶形畸变和枕形畸变，数学中常用泰勒公式展开来进行建模，这里会引入 k1,k2,k3 等径向畸变的参数，公式为

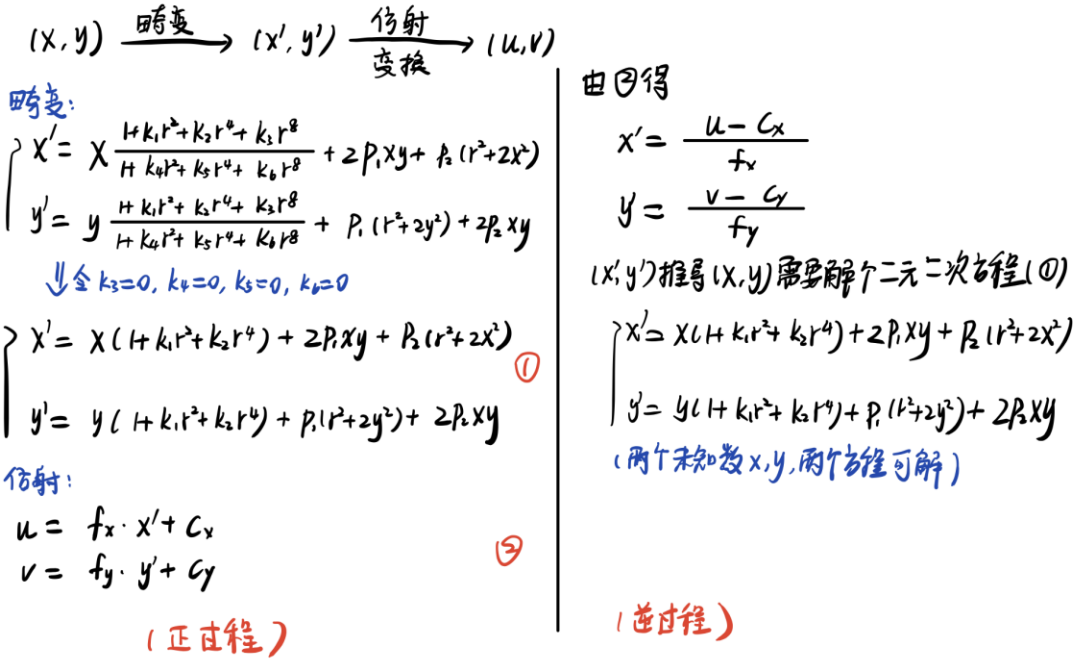
$$x_0 = x(1 + k_1r^2 + k_2r^4 + k_3r^6)$$

$$y_0 = y(1 + k_1r^2 + k_2r^4 + k_3r^6)$$

   - **切向畸变**主要是镜头与像平面不平移造成的，同样的数学上引入 p1、p2 两个参数来描述：

$$x_0 = 2p_1xy + p_2(r^2 + 2x^2) + 1$$

$$y_0 = p_2(r^2 + 2y^2) + 2p_2xy + 1$$



2.

References:

- 博客园|第六节、双目视觉之相机标定
- opencv-doc|camera*calibration*and*3d*reconstruction

Q5: Describe what the camera calibration does

相机标定 **camera calibration** 即通过一系列照片来求出相机的内参数、外参数、畸变参数的过程；

相机标定是计算机视觉的一个基础工作，为图片的正畸、测距做准备

Q6: Programming for camera calibration with opencv

Ans6:

单目相机标定代码地址
https://github.com/QYHcrossover/Project*Stereo/blob/master/single*calibration.py

刚开始直接看 Opencv 的文档有些吃力，主要让我迷糊的是 API 中的输入和输出对应的是什么，在调用 calibrateCamera 前应该做哪些预备工作等；于是我就找到了 CSDN 上的一篇博客，先搞清楚了算法的流程：

1. 准备标定图片
2. 对每一张标定图片，提取角点信息
3. 对每一张标定图片，进一步提取亚像素角点信息
4. 在棋盘标定图上绘制找到的内角点（非必须，仅为了显示）
5. 相机标定

然后这个博客非常详细，附带了 python 代码和代码解析；在对源码简单修改后，我便跑通了老师您给的测试图片，成功地标定了相机的内参数、外参数和畸形矩阵；然后我便对应 API 逐行理解每行代码，通过 jupyter 逐行输出中间结果，理解起来也没什么太大的障碍；

唯一有些疑问的是最后计算出的外参数中的 **R 旋转系数**和 **T 平移系数**，照道理每个图片单应矩阵中的 R 应该是[R1，R2],是个 3 * 2 的二维矩阵，计算出的 R 和 T 一样居然是 3 * 1 的向量，~~这让我有所不解。猜测 R1 和 R2 彼此正交，且 R1R2 模都为 1，R1 定了以后 R2 也定了（不知道对不对）~~

前面猜测完全错了，后面在双目视觉的时候又回过头看感觉不对，又查了查资料，原来这个向量表示的是 3 的维的旋转弧度，而不是旋转矩阵中的一个向量！opencv 也有专门的函数 `Rodrigues()`来进行旋转弧度和旋转向量的转换，所以我顺手额外地求了下外参矩阵；所以我将代码整合封装了一下，也为后面的双目标定做准备。

Reference：

- opencv-doc|camera*calibration*and*3d*reconstruction
- CSDN|Python-OpenCV 相机标定、张正友标定法（主要是这篇，写得很详细）
- CSDN|三维坐标变换——旋转矩阵与旋转向量
- CSDN|旋转向量和旋转矩阵的互相转换（python cv2.Rodrigues()函数）
- opencv-doc|calibrateCamera()
- opencv-doc|Rodrigues()

Q7: Undistort the images

对图片去畸变这个在我刚才看到的博客中也提到了，用的 opencv 的函数是 `cv2.undistort()`

```
dst = cv2.undistort(src, cameraMatrix, distCoeffs[, dst[, newCameraMatrix]]  )
```

或者也可以用 `cv2.initUndistortRectifyMap` 和 `cv2.remap` 先计算一个先计算一个从畸变图像到非畸变图像的映射，然后使用这个映射关系对图像进行去畸变。

```
# undistort
mapx,mapy = cv2.initUndistortRectifyMap(mtx,dist,None,newcameramtx,(w,h),5)
dst = cv2.remap(img,mapx,mapy,cv2.INTER_LINEAR)
```

单相机标定和畸变校正这块原先是有一个 BUG，校正后图像会发生扭曲；起初并没有在意，到了后面立体校正发现结果不对才最后找到这，原因是 w 和 h 写反了

References:

- opencv-doc|undistort()

Q8: Programming for Zhang's method

代码地址:https://github.com/QYHcrossover/Project_Stereo/tree/master/Zhang's%20method

自己实现张正友算法应该来说是挺复杂的，看了博客理解了算法的大致流程；

Algorithm 张正友平面标定法

1: 初始化：摄像机位置保持不变，将标定板置于不同位置采集 m 幅图像（世界坐标系/单应矩阵随之变化），检测图像中的角点，设每幅图像中可检测出 n 个角点
2: **for** $j \in [1, m]$ **do**
3: 单应矩阵求解：求解方程 $AH_j = 0$（共 2n 个方程，9 个未知数），得到单应矩阵 H_j 的初值，并考虑噪声影响，进行非线性优化
4: **end for**
5: 摄像机内参数求解：综合 m 幅图像中的单应矩阵 Hj，求解方程 $vb = 0$（共 2m 个方程，6 个未知数），将中间矩阵 B 进行分解，获得内参数矩阵 K（5 个参数）
6: **for** $j \in [1, m]$ **do**
7: 摄像机外参数求解：计算 R_j 和 t_j
8: **end for**
9: 畸变系数求解：综合 m 幅图像中的 n 个角点，求解方程 $Dk = d$（共 2mn 个方程，2 个未知数），得到畸变系数
10: 参数最优化：通过非线性最小二乘法，优化 K（5 个参数），R_j 和 t_j（6m 个参数）和 k_1, k_2

一共涉及到**单应性矩阵求解**、**内参数求解**、**外参数求解**，**畸变系数求解**；用到的方法涉及到 SVD 分解，最小二乘法，极大似然估计等；

确实非常复杂，我估计自己实现需要很长时间；不过我找到一份源码，经过实测可以运行；

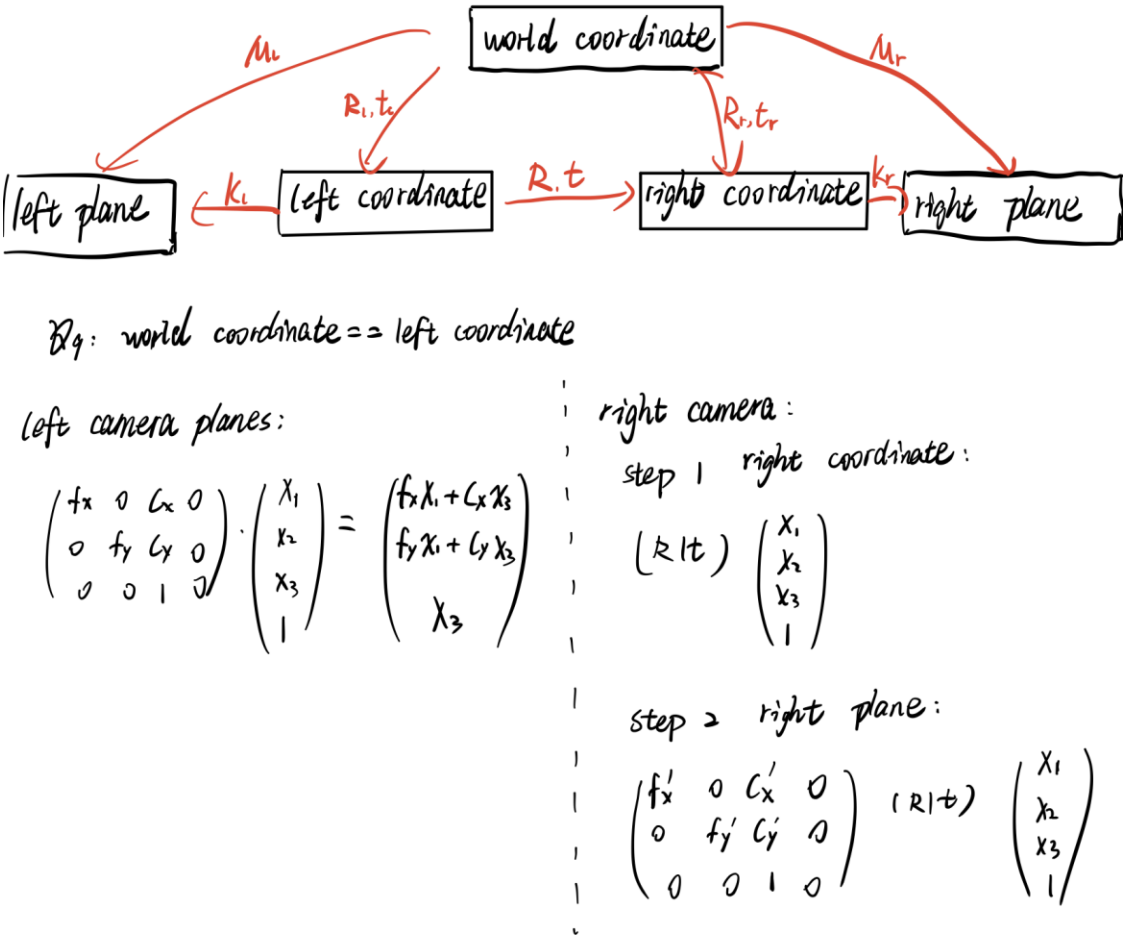
结果和 opencv 实现的非常接近，还是由于时间关系也没有对源码进行深入研究。

References:

- CSDN|【MQ 笔记】张氏标定法学习笔记：A Flexible New Technique for Camera Calibration 详解
- 知乎|【机器视觉】张氏法相机标定
- 知乎|相机标定之张正友标定法数学原理详解
- CSDN|张正友相机标定法原理与实现

- CSDN|张正友标定算法原理详解

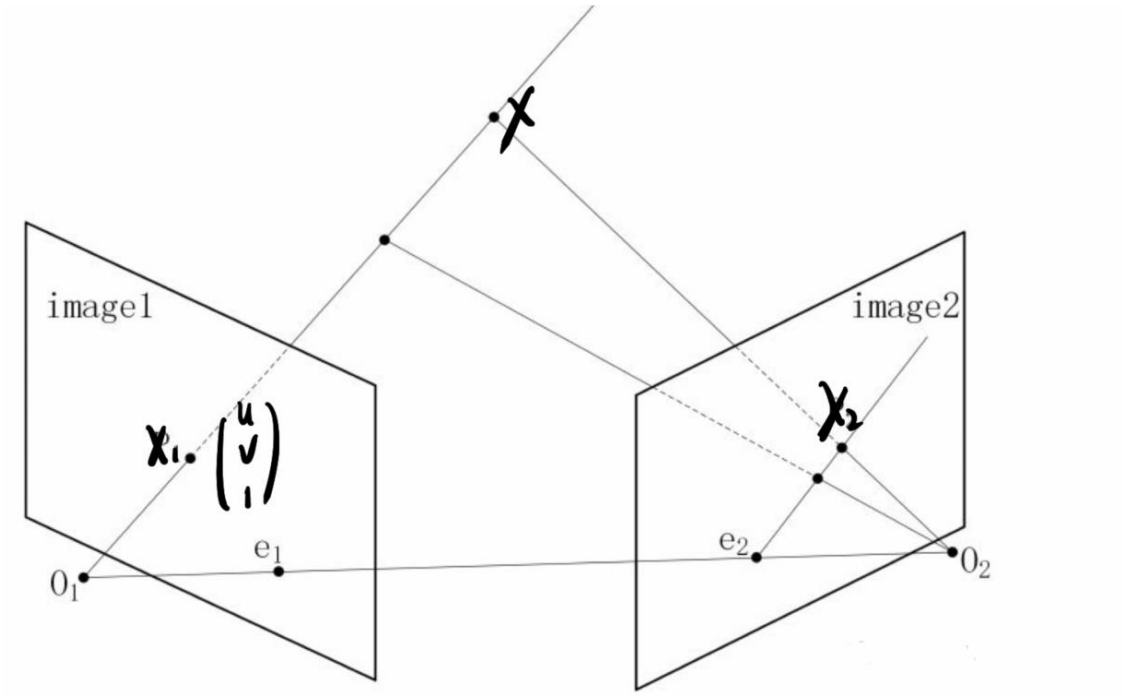
Q9: Projection



画了张转换图瞬间就清晰了很多，这道题先前漏了过去后来补做的，题目的关键点还是在于理解各个坐标系的转换过程。

Q10: Epipolar Line

Ans10：



1. left pixel plane → left camera coordinates

$M_1 \cdot C_{L1} = X_1$

$\Rightarrow \quad C_{L1} = M_1^{-1} X_1 \quad \cdots\cdots\cdots$ 公式1

（C_{L1} 代表在左相机坐标系下 X_1 点的坐标）

2. left camera coordinates → right camera coordinates

$C_{R1} = R C_{L1} + t$ （C_{R1} 代表在右相机坐标系下该点的坐标）

$= R(M_1^{-1} X_1) + t$

3. $O_1X_1O_2$ 平面的方程：

法向量：$\overrightarrow{O_1O_2} \times \overrightarrow{O_2X_1} = t \times \left(R(M_1^{-1} X_1) + t\right)$

平面方程：$C_{R2} \cdot (\overrightarrow{O_1O_2} \times \overrightarrow{O_2X_1}) = 0$ （C_{R2} 代表在右相机体系下点2的坐标）

又因为 $M_2 C_{R2} = X_2$

$\therefore \quad C_{R2} = M_2^{-1} X_2$

直线方程为：$M_2^{-1} X_2 \cdot \left(t \times (R(M_1^{-1} X_1) + t)\right) = 0$

其中 M_1，M_2 题中给出，X_2 为未知点坐标，R，t 分别为旋转矩阵，平移向量

X_1 为原始点坐标 $X_1 = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$

这道题做得有点虚，也不知道对不对；首先我看了资料理解了这个直线就是图中的e_2x_2,然后开始求它方程；

主要还是坐标系太混乱了，涉及到左相机的立体坐标系、像素坐标系以及右相机的立体坐标系、像素坐标系等；所以我的思路是统一坐标系，因为题目中求的是在**右相机体系下的直线方程**，所以都需要**统一到右相机的坐标体系**

像素坐标和相机坐标之间的转换挺熟悉了，从左到右的转换就第一次尝试（不知道有没有写反）；这边主要的难点是**平移向量 T 的几何意义**，实际上对应了O_1O_2,这点挺难发现的（不知道对不对）；另外这里还涉及到了**求平面方程**的知识，包括法向量怎么求（向量叉乘）等等，加上我不是完全理解，所以在第 3 步求平面方程那块C_{r2}那边不知道要不要减去一个 **t**，知乎上写的是直接C_{r2},我主观觉得两者皆可，都是该平面的向量；

看完 tutorial 2 再来当初求极线方程，感觉大致上写的是对的；也可以在求平面方程时，X_1和X_2的位置换一下；就是原本是$O_2X_2 \cdot (O_1O_2 \times O_2X_1)$,也可以写成$O_2X_1 \cdot (O_1O_2 \times O_2X_2)$，形式上可能会不一样，不过最终结果是一样的；也没有把中间的结果写成本征矩阵或者基础矩阵。

References:

- 知乎|小白视角之极线约束 我觉得这篇挺好理解的
- https://courses.engr.illinois.edu/cs543/sp2011/lectures/Lecture%2023%20-%20Epipolar%20Geometry%20and%20Stereo%20-%20Vision_Spring2011.pdf 老师给的 Epipolar tutorial2，这个讲义太棒了

Q11: Fundamental Matrix

求解基础矩阵，这道题的重点和难点还是**坐标体系转换**和求**平面方程**的问题，和上个题目求极线约束类似；从我看到的资料和以我的理解上看，这题的坐标体系应该是要全部统一成**左相机的坐标系**，这个和上一题恰恰相反；

博客中有两处让我大为不解；首先在第二节中，给出P_l和P_r的关系是$P_r = RP_l + T$从左到右的转换顺序，然后到第三节给出的P_l和P_r的关系是：$P_r = R(P_l - T_r)$,那可以得出T和T_r的关系，有 $T = -RT_r$（不知道是否正确）

最让我奇怪的是公式 8 $P_r^T RSP_l = 0$,按照推导来说这式子应该是R^{-1}啊，所以到底是$P_r = R(P_l - T_r)$错了还是这个式子错了呢？？

后面看了老师给的 tutorial 2,思路瞬间清晰了起来啊！终于理解为什么有极线约束，本征矩阵和基础矩阵到底是怎么来的；**写了篇博客**记录了下这些概念和推导过程，基础矩阵的推导也在这里面，这里就不重复写了；博客链接 https://blog.csdn.net/weixin_43796012/article/details/107547525，写博客写得还是挺累的

References:

- https://courses.engr.illinois.edu/cs543/sp2011/lectures/Lecture%2023%20-%20Epipolar%20Geometry%20and%20Stereo%20-%20Vision_Spring2011.pdf 老师给的 Epipolar tutorial2，这个讲义太棒了

- CSDN|机器视觉学习笔记（6）——双目摄像机标定参数说明

Q12: Stereo Calibration

代码地址 https://github.com/QYHcrossover/Project*Stereo/blob/master/stereo*calibration.py

这道题被老师标了两星，看了老师给的资料感觉并不难，就是双目标定的基本内容，在两个单目标定的基础上增加了 **R 和 L**，即两个相机坐标之间的转换关系；步骤也很清晰：

1. 先各自求单个相机的内参和外参,分别记作$(R_l,t_l),(R_r,t_r)$
2. 根据下面的式子求解 R 和 L

综合（1）（2）两式，可以推得：

$$\begin{cases} R = R_r R_l^T \\ T = T_r - RT_L \end{cases} \tag{3}$$

单目标定中相机外参数就是此处的R_l，T_l，R_r和T_r，带入（3）式就可以求出R和T。

后面看了其他博客忽然发现 opencv 也有个专门做双目相机标定的 API，不需要我们自己手动算旋转矩阵然后用公式自己算，另外还返**回了本征矩阵、基础矩阵**等参数！自己算还有一个缺陷就是上面公式中的 R 和 T 是**从左到右的旋转矩阵和平移向量**，自己算出来的未必准确何况后续立体校正还需要用到双摄像的标定的结果，所以是还是老老实实用 opencv 的 API；后面我又进行了一下代码重构，将双摄像机标定过程封装成函数，以便后面调用。

References:

- CSDN|机器视觉学习笔记（6）——双目摄像机标定参数说明
- opencv-doc|stereoCalibrate()

Q13: derive and calculation

用F表示基础矩阵，那么有

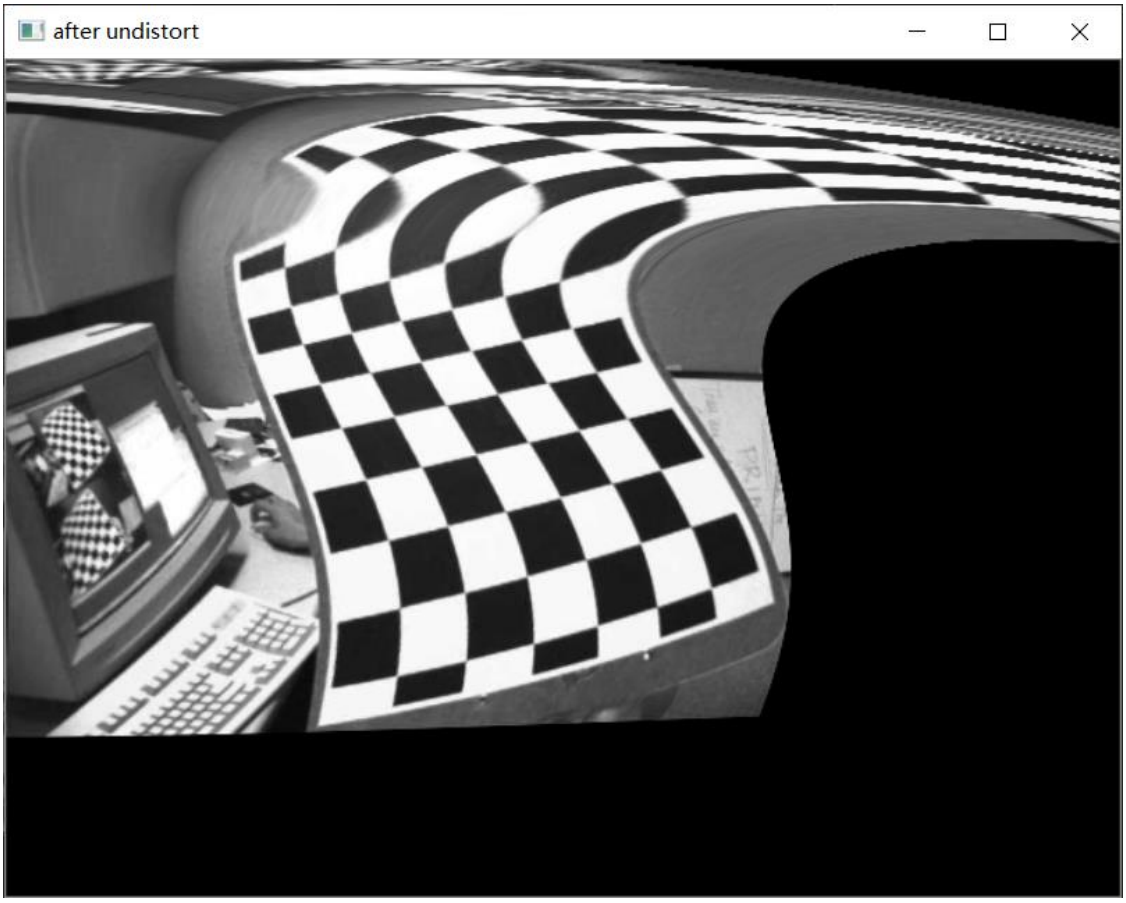
$$p_l^T F p_r = 0$$

该方程就是极线方程，p_r就是右相机平面上对应点所在的位置

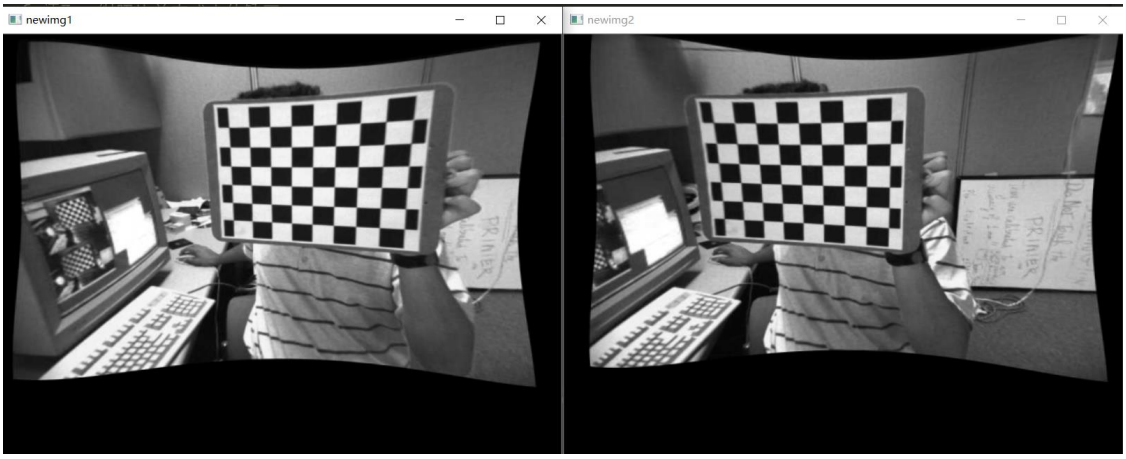
该点的 3D 坐标是，$M_r^{-1} P_r$

Q14: Rectification

因为我用的是 python 的 opencv，找了下网上现成的基本上没有；所以我只好慢慢地抠 python-opencv 文档了，好在整体逻辑是一样的，python 的 API 还更加简洁。摸索地完成了大概流程后，**显示矫正好的图像居然一片漆黑**。其实心里一直比较虚，因为之前在单相机矫正的时候，结果居然是扭曲了。



所以我觉得源头还是在单相机那块，找了半天 BUG，和参考代码一一比对，居然发现图像的横纵坐标在函数中填反了！！天啊，改了这个 BUG 后，undistort 的结果也正常了,再重新运行 rectification 代码，成功了！如下图所示，没有做裁切。

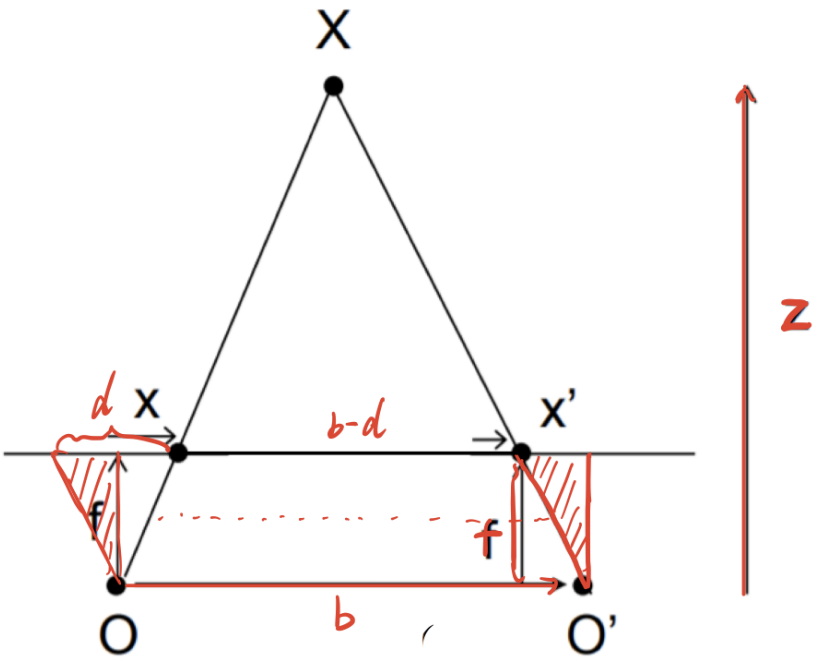


References:

- CSDN|双摄像头立体成像(三)-畸变矫正与立体校正
- 博客园|第七节、双目视觉之空间坐标计算
- CSDN|StereoRectify()函数定义及用法畸变矫正与立体校正
- opencv-doc|stereoRectify()
- opencv-doc|initUndistortRectifyMap()
- opencv-doc|remap()

Q16: Depth-Disparity

tutorial 2 的一张图画得，想到了初中学的相似三角形；需要稍微动点脑筋的是d的含义，其实也很简单补块三角形就看出来了，如图所示：



相似三角形嘛，$\Delta Xxx' \sim \Delta XOO'$，有比例关系

$$\frac{b-d}{b} = \frac{z-f}{z}$$

$$1 - \frac{d}{b} = 1 - \frac{f}{z}$$

$$z = \frac{bf}{d}$$

References:

- https://courses.engr.illinois.edu/cs543/sp2011/lectures/Lecture%2023%20-%20Epipolar%20Geometry%20and%20Stereo%20-%20Vision_Spring2011.pdf 老师给的 Epipolar tutorial2，这个讲义太棒了

Q17: SGBM

代码地址: https://github.com/QYHcrossover/Project_Stereo/blob/master/SGBM.py

简单地理解了一下 SGBM 的核心思想，通过选取每个像素点的视差 d，组成一个 disparity map，设置一个和 disparity map 相关的全局能量函数，使这个能量函数最小化，以达到求解每个像素最优 disparity 的目的。

$$E(D)=\sum_{p}\left(C\left(p,D_p\right)+\sum_{q\in N_p}P_1 I\left[\left|D_p-D_q\right|=1\right]+\sum_{q\in N_p}P_2 I\left[\left|D_p-D_q\right|>1\right]\right)$$

对于灰度差值等于 1 的施加一个比较小的惩罚P_1,对于灰度差值大于 1 的施加一个比较大的惩罚P_2;至于怎么求最优解的话，SGM 将优化问题分解为四条路径上的动态规划问题。

至于 opencv 实现，找到一份比较靠谱的参考代码，加上 opencv 的文档，实现 SGBM 总体上没有遇到什么困难。

References:

- CSDN|SGM 算法思路
- CSDN|【码上实战】【立体匹配系列】经典 SGM：（1）框架与类设计
- CSDN|Python/openCV 中 cv2.StereoSGBM_create()参数的含义
- CSDN|python 实现 SGBM 图像匹配算法

在网上找立体匹配资料的时候，我找到了一家做 3D 视觉的公司奥比中光；它对外有一次技术讲座，就是将立体匹配的发展，我看了这个视频了解了立体匹配的很多概念，包括：

- 立体匹配的**难点**
- 立体匹配**算法分类**（局部、半局部、全局方法）
- 立体**匹配流程**（**匹配代价计算、代价空间、代价聚合、视差计算、视差优化**）
- 有哪些端到端的方法

写了一篇博客记录了学到的东西，博客链接:立体匹配算法笔记

References:

- 奥比中光直播课

Q18: Unsupervised

我觉得有监督的深度学习算法**不太容易直接用到双目视觉匹配上面去**，原因有两点：

- 第一，有监督深度学习**高度依赖于训练数据**，而由于往往训练数据是特定场景的（比如 Kitti 2012/2015 数据集是外景，Middlebury Stereo 数据集是单纯室内场景）;**训练数据和真实的测试环境的差异大**，模型相比于传统算法**泛化性能差**
- 第二，监督学习对训练数据(无论是数量还是质量上)要求都很高，而据我了解获取深度图或视差图（也就是标签）是一件不太容易的事情；像 **LiDAR** 这样的设备是极其笨重且昂贵的，它所能收集的只是稀疏的深度信息，而我们需要的是**密集的深度图（dense depth map）**；而基于结构光的设备往往只能**在室内场景**下进行深度信息标注，在室外场景下难以达到较高的标注质量。

然后我也简单看一篇介绍双目匹配中的自监督学习的方法，理解了其中的几个思路：

- **充分利用左右两张图片的信息**，设计一个网络以左图像为直接输入，直接输出预估的视差；利用右图与视差重构出左图，然后定义一种损失函数估计重构出的 z 左图和真实左图之间的差距，最小化这个差距
- 和上述类似，左图和右图都作为输入，直接输出两图的视差，定义一种**左右视差一致性损失**，最小化该损失
- 利用一些传统算法，在传统算法输出的视差图上构建网络输出其**视差置信度**，挑选视差置信度高的像素位置来做训练

具体的网络结构、损失函数的设计等细节还没有来得及看，老实说我无监督学习和自监督学习这两方面的东西看得太少，缺乏一些整体的把握；希望自己有时间好好加强一下这方面知识。

References：

- 知乎|双目深度估计中的自监督学习概览

Q21: Single image depth estimation

用单张图片做深度估计？并且是深度学习方法？

拿到这道问题，我的第一想法很有意思，就是拿人眼做类比；人类获取的也是视网膜上的 2D 图像，可我们依然能感知物体与我们的距离。人可以那相机+深度学习依然能行啊；可等等，人不是两只眼睛吗，那不是双目视觉了吗？

……

人闭一只眼看世界也能行啊！不过又转念想想，我们人依靠先验知识也只能简单判断下物体的大概距离，更何况还有“视觉欺骗”这东西。而计算机想依靠单张图片来计算一个精确的深度确实不是一件容易的事情。

好了回归正题，如何构建一个神经网络来估计深度；我的一个最朴素的思想是构建一个端到端的网络，图片直接输入到网络中去；通过各种操作最后输出一个相同尺寸的二维张量，每个像素所在的值代表深度距离；相当于一种深度回归网络？

这个过程需要先验信息，因为图片与图片之前场景差异太大了；如果室内的话最远也不过几米，而室外的话就远远不止了！

看了一篇博客讲单目视觉深度估计的发展，了解到了很多传统方法，可以靠很多参考物比如(霾、阴影、表面纹理、遮挡、消失点)等；

深度学习方面，也有各种思路：

- 可以将深度估计与语义分割相结合
- 可以当做一种搜索问题，假设相似图像具有相似的深度图
- 也利用视频中前后两帧做光流
- 也有和我的最初想法相似的，先进行下采样后进行上采样；在此基础上做各种优化

也算对单目视觉深度估计有了一个最初的直观的认识，感觉这个领域目前还在研究阶段；比起图像识别、目标检测这些更加容易出成果。

References:

- 知乎|单目视觉深度估计测距的前生今世
- 知乎|基于深度学习的单目深度估计综述

Q22: More methods

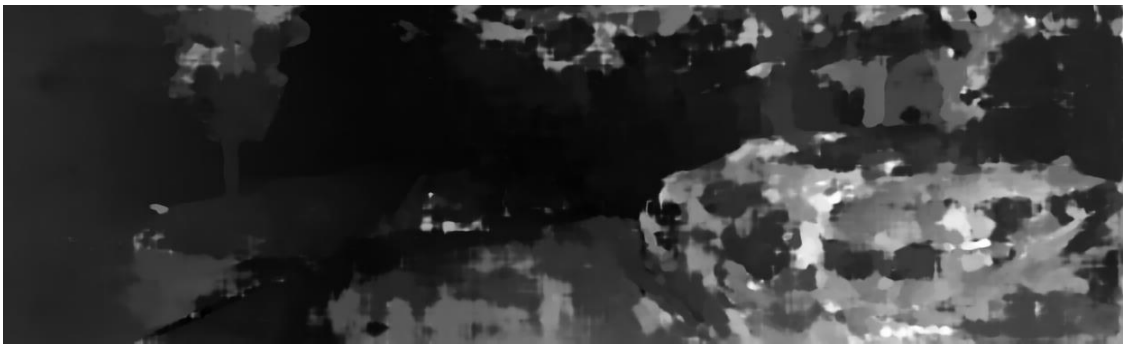
本来想找 KITTI 上最高级的方法来跑一下的，无奈 TOP1 没有开源，开源的排名最靠前的算法看了下计算要求，GPU 显存不低于 11GB，遂放弃。

然后就选了 PMS-net 这个经典的网络；GIthub 拷贝了源码，下载了 20MB 的预训练模型，下载了 Kitti 的 1.56GB 的 data*scene*flow 数据集；然后在 Kitti 测试集中的两张图片测试，结果发现自己的显存还是不够，哎！

```
    input = module(input)
  File "D:\Anaconda3\lib\site-packages\torch\nn\modules\module.py", line 550, in __call__
    result = self.forward(*input, **kwargs)
  File "D:\Anaconda3\lib\site-packages\torch\nn\modules\conv.py", line 933, in forward
    output_padding, self.groups, self.dilation)
RuntimeError: CUDA out of memory. Tried to allocate 594.00 MiB (GPU 0; 2.00 GiB total capacity; 943.92 MiB already allocated;
113.17 MiB free; 1.12 GiB reserved in total by PyTorch)

H:\project\python\DL\PSMnet\PSMNet>
```

登上了自己常用的 GPU 云服务器，简单配置下，修改了几个小 bug，终于跑起来了，效果如下：

距离最近的右侧的几辆车灰度最浅，这个估计得还不错；只是图片中间（更远的景物）距离的层次信息不太明显，就黑漆漆的一片。

因为时间的原因就没有来得及尝试更多的深度学习方法了，至于双目视觉都有的劣势是计算成本都很高吧.

References：

- github|PSMNet
- CSDN|运行 PSMNet 网络时遇到的问题及解决方案
- CSDN|论文阅读笔记之 Dispnet